The Journal of Systems and Software 85 (2012) 1346–1362

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Mining frequent patterns from dynamic data streams with data load management☆

Chao-Wei Li, Kuen-Fang Jea*, Ru-Ping Lin, Ssu-Fan Yen, Chih-Wei Hsu

Department of Computer Science and Engineering, National Chung-Hsing University, 250 Kuo-Kuang Road, Taichung 40227, Taiwan, ROC

ARTICLE INFO

Article history:
Received 1 November 2010
Received in revised form 8 August 2011
Accepted 13 January 2012
Available online 24 January 2012

Keywords:
Data mining
Data streams
Frequent patterns
Combinatorial approximation
Overload handling
Load shedding

ABSTRACT

In this paper, we study the practical problem of frequent-itemset discovery in data-stream environments which may suffer from data overload. The main issues include frequent-pattern mining and data-overload handling. Therefore, a mining algorithm together with two dedicated overload-handling mechanisms is proposed. The algorithm extracts basic information from streaming data and keeps the information in its data structure. The mining task is accomplished when requested by calculating the approximate counts of itemsets and then returning the frequent ones. When there exists data overload, one of the two mechanisms is executed to settle the overload by either improving system throughput or shedding data load. From the experimental data, we find that our mining algorithm is efficient and possesses good accuracy. More importantly, it could effectively manage data overload with the overload-handling mechanisms. Our research results may lead to a feasible solution for frequent-pattern mining in dynamic data streams.

1. Introduction

Nowadays many commercial applications have their data presented in the form of continuously transmitted stream, namely *data streams*. In such environments, data is generated at some end nodes or remote sites and received by a local system (to be processed and stored) with continuous transmission. It is usually desirable for decision makers to find out valuable information hidden in the stream. Data-stream mining is just a technique to continuously discover useful information or knowledge from a large amount of running data elements. Like data mining in traditional databases, the subjects of data-stream mining mainly include *frequent itemsets/patterns*, *association rules* (Agrawal and Srikant, 1994), *sequential rules*, *classification*, and *clustering*. Through the data-stream mining process, knowledge contained inside the stream can be discovered in a dynamic way.

Data mining from data streams has three kinds of *time models* (or *temporal spans*) (Zhu and Shasha, 2002). The first one is *landmark window model*, in which the range of mining covers all data elements that have ever been received. The second one is *damped/fading window model*, in which each data element is associated with a variable weight, and recent elements have higher weights than previous ones. The third one is *sliding window model*, in which a fixed-length window which moves with time is given, and the range of mining covers the recent data elements contained within the window. Due to the fact that early received data elements may become out of date and/or insignificant, i.e., the *timeliness* factor, among the three models, the sliding window model is more appropriate for many data-stream applications such as finance, sales, and marketing.

A data-stream mining system may suffer the problem of *data overload*, just like the case of over-demand electricity to a power supply system. A data stream is usually dynamic and its running speed may change with time. When the data transmission rate of the data-stream source exceeds the data processing rate of the mining algorithm of a mining system, e.g., during a peak period, the system is *overloaded with data* and thus unable to handle all incoming data elements properly within a time-unit. Furthermore, an overloaded system may work abnormally or even come into a crash. Accordingly, it is essential for a data-stream mining system to adequately deal with data overload and/or spikes in volume.

In this paper, we propose a load-controllable mining algorithm for discovering frequent patterns in transactional data streams. The mining algorithm works on the basis of *combinatorial approximation* (Jea and Li, 2009). To address the possible case of peak data-load at times, two dedicated overload-handling mechanisms are designed for the algorithm to manage overload situations with their respective means. With the load-controllable ability, a mining system with the proposed algorithm is able to work normally during

☆ This research is supported in part by NSC in Taiwan, ROC under Grant No. NSC 98-2221-E-005-081.

* Corresponding author. Tel.: +886 4 22840497x906; fax: +886 4 22852396.
E-mail address: kfjea@cs.nchu.edu.tw (K.-F. Jea).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2012.01.024

high-data-load periods. Besides, according to our experimental data, the mining results (after overload handling) still possess reasonable quality in term of accuracy.

The rest of this paper is organized as follows. In Section 2, related work regarding data-stream frequent-pattern mining and data-overload handling is described. Section 3 gives the symbol representation, problem definition, and goal of this research. In Section 4, a mining algorithm together with two overload-handling mechanisms is proposed and explained in detail. Section 5 presents the experimental results with analyses. In Section 6, discussions are given on the overload-handling mechanisms. Finally, Section 7 concludes this work.

2. Related work

Frequent-pattern mining from data streams is initially limited to *singleton items* (e.g., Charikar et al., 2004; Fang et al., 1998). Lossy Counting (Manku and Motwani, 2002) is the first practical algorithm to discover frequent itemsets from transactional data streams. This algorithm works under the landmark window model. In addition to the minimum-support parameter (*ms*), Lossy Counting also employs an error-bound parameter, ε, to maintain those infrequent itemsets having the potential to become frequent in the near future. Lossy Counting processes the stream data batch by batch, and each batch includes bucket(s) of transactions. With the use of parameter ε, when an itemset is newly found, Lossy Counting knows the upper-bound of count that itemset may have before it has been monitored. As a result, the error in itemset's frequency count is limited and controllable. According to the experiments conducted by Manku and Motwani (2002), Lossy Counting can effectively find frequent itemsets over a transactional data stream. This algorithm is a representative of the *ε-deficient mining* methods and has many related extensions. For example, based on the estimation mechanism of Lossy Counting, a sliding window method for finding recently frequent itemsets in a data stream is proposed by Chang and Lee (2004).

A different type of method is to process on stream elements within a limited range and offer no-false mining answers. DSTree (Leung and Khan, 2006) is a representative approach of one such type, which is designed for *exact (stream) mining* of frequent itemsets under the sliding window model. Given a sliding window, DSTree uses its tree structure for capturing the contents of transactions in each batch of data within the current sliding window. More specifically, DSTree enumerates all itemsets (of any length) having ever occurred in transactions within the current window and maintains them fully in its tree structure. The update of tree structure is performed on every batch, while the mining task is delayed until it is needed. According to the experimental results given by Leung and Khan (2006), mining from DSTree achieves 100% accuracy (since all itemsets having ever occurred are stored and monitored). However, because this method needs to enumerate every itemset in each of the transactions, its efficiency is badly affected by the great computation complexity. As a result, it can hardly manage with a data stream consisting of long transactions.

Besides DSTree, there are other methods belonging to the type of exact stream mining. Li and Lee (2009) proposed a bit-sequence based, one-pass algorithm, called MFI-TransSW, to discover frequent itemsets from data-stream elements within a transaction-sensitive sliding window. Every item of each transaction is encoded in a *bit-sequence* representation for the purpose of reducing the time and memory necessary for window sliding; to slide the window efficiently, MFI-TransSW uses the *left bit-shift* technique for all bit-sequences. In addition, Tanbeer et al. (2009) proposed an algorithm called CPS-tree, which is closely related to DSTree, is proposed to discover the recent frequent patterns from a data stream over a sliding window. Instead of maintaining the batch information (of the sliding window) at each node of the tree structure (as DSTree does), CPS-tree maintains it only at the last node of each path to reduce the memory consumption. The authors also introduce the concept of dynamic tree restructuring to produce a compact frequency-descending tree structure during runtime.

Another type of method is to perform the mining task, i.e., discover frequent itemsets, through a support approximation process. One feasible approach is to apply the idea of *Inclusion–Exclusion Principle* in combinatorial mathematics (Liu, 1968), whose general form is shown in Eq. (1), to data-mining domain for *support calculation*, and further apply the theory of *Approximate Inclusion–Exclusion* (Linial and Nisan, 1990) for *approximate-count computation*. This mining approach is called *CA* (*combinatorial approximation*). The DSCA algorithm (Jea and Li, 2009) is a typical example of such an approach. DSCA operates under the landmark window model and monitors all itemsets of lengths 1 and 2 existing in the data stream. It performs the mining task by approximating the counts of longer itemsets (based on the monitored itemsets) and returning the frequent ones as the mining outcome. According to the experimental data given by Jea and Li (2009), the performance of DSCA is quite efficient. Besides DSCA, the SWCA algorithm (Li and Jea, 2011) is also an example of CA-based approach. This method is under the sliding window model and has made an effort to improve the accuracy of support approximation.

$$|A_1 \cup A_2 \cup \ldots \cup A_m| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| + \cdots + (-1)^{m+1} |A_1 \cap A_2 \cap \ldots \cap A_m| \quad (1)$$

In the electrical utility industry, *load shedding*, an intentionally engineered electrical power outage, is a last resort measure used by an electric utility company in order to avoid a total blackout of the power system. It is usually in response to a situation where the demand for electricity exceeds the power supply capability of the network. In the data-stream mining domain, many data-stream sources are prone to dramatic spikes in volume (Babcock et al., 2003). Because the peak load during a spike can be orders of magnitude higher than normal loads, fully managing a data-stream mining system to handle the peak load is almost impractical. Therefore, it is important for mining systems which process data streams to be adaptable to unanticipated variations in data transmission rate. An overloaded mining system is unable to manage on its own with all incoming data promptly, so *data-overload handling*, or simply *overload handling*, such as discarding some unprocessed data (i.e., load shedding) becomes necessary for the system to be durable.

The Loadstar system proposed by Chi et al. (2005) introduces load shedding techniques to *classifying* multiple data streams of large volume and high speed. Loadstar employs a metric known as the *quality of decision* (QoD) to measure the level of uncertainty in classification. When sources are in competition with each other for resources, the resources are allocated to the source(s) where uncertainty is high. In order to make optimal classification decisions, Loadstar relies on *feature prediction* to model the dropped and unseen data, and thus can adapt to the changing characteristics in data streams. Experimental results show that Loadstar offers a nice solution to data-stream classification with the presence of system overload.

To our knowledge, there are no representative studies concerning overload-handling schemes/mechanisms for frequent-pattern mining in data streams so far. However, the work conducted by Jea et al. (2010) has made a preliminary attempt. In the paper, a data-stream frequent-itemset mining system is proposed where the mining algorithm is a Lossy Counting based method. The system finds frequent itemsets from the data within a sliding window.